Some OOP stuff

@dave1010

“Object Calisthenics”

Jeff Bay - Thoughtworks Anthology

Don’t follow this blindly*

*give it a go

1. Only One Level Of Indentation Per Method

```
public function bar()
{
        if ($this->wop) {
                for ($i = 0; $i < $this->bloops; $i++) {
                        if ($this->has($i)) {
                                $this->barg($i);
         }
      }
  }
}
```

1. Only One Level Of Indentation Per Method

```
public function bar()
{
    if (!$this->wop) {
        return; // return early
    }
    for ($i = 0; $i < $this->bloops; $i++) {
        $this->tryBarg($i); // extract method
    }
}
```

2. Don’t use the else keyword

– – –

```
if ($this->wop) {
    $this->doWop('wop');
} else {
    $this->doNotWop('not');
}
```

2. Don’t use the else keyword

```
if ($this->wop) {
    $this->doWop(‘wop’);
    return; // return early
}
$this->doNotWop(‘not’); // no indentation, less to think about
```

2. Don’t use the else keyword

```
$wop = $this->translate($this->wop); // introduce variable

$this->wopOrNot($wop);
```

3. Wrap All Primitives And Strings

```
    $time = '23:59';

    $this->clock->travelTo($time);

...

$hhmm = explode(':', $time); // this is all over the place

echo $hhmm[0] * 3600 + $hhmm[0] * 60;
```

3. Wrap All Primitives And Strings

```
// logic is in 1 place and has tests!


$time = Time::fromHoursAndMinutes(23, 59);

echo $time->secondsSinceMidnight();
```

4. First Class Collections

```
class Schedule

{

    private $bookings = []; // does stuff with these and loads of other stuff

    private $timeZone;

    private $contact;

    public function paidBookings() { … }

}
```

4. First Class Collections

```
class BookingCollection
{
    private $bookings = [];
}
```

5. One Dot (->) Per Line

```
$this->order->getUser->getProfile()->getAddress();

// User::getAddress() is called all over app

// Google "Law of Demeter"
```

5. One Dot (->) Per Line (after $this)

```
// $this doesn't have to know about User
$this->order->getShippingAddress();

class Order
{
    public function getShippingAddress() {}
}
```

6. Don’t Abbreviate

———

```
$name = e($name); // Don’t like typing “encodeForHtml” all the time
$address = e($address);
$dob = e($dob);
```

6. Don’t Abbreviate

———

```
$name = encodeForHtml($name);
$address = encodeForHtml($address);
$dob = encodeForHtml($dob);
```

6. Don’t Abbreviate

———

```
renderTemplate($template, compact($name, $address, $dob));

// wanting to abbreviate = lots of duplication
```

7. Keep All Entities Small

< 100 lines

8. No Classes With More Than Two Instance Variables

```
class Booking
{
    private $itemName;
    private $itemType;
    private $itemId;
    private $userEmail;
    private $userName;
}
```

// “What properties can I chuck in this object?”

8. No Classes With More Than Two Instance Variables

```
class Booking

{

    private $item;

    private $user;

}

// How is this object composed?
```

9. No Getters / Setters / [public] Properties

```
function transfer(Booking $booking, User $toUser) {

    $user = $booing->getUser();

    $user->setEmail($toUser->getEmail());

}

// Google “Open/Closed Principle” and “Tell, Don't Ask”
```

9. No Getters / Setters / [public] Properties

```
class Booking

{

    public function transferTo(User $user) {}

}
```

Summary

---

1. Only One Level Of Indentation Per Method
2. Don’t Use The Else Keyword
3. Wrap All Primitives And Strings
4. First Class Collections
5. One Dot Per Line
6. Don’t Abbreviate
7. Keep All Entities Small
8. No Classes With More Than Two Instance Variables
9. No Getters/Setters/Properties